IN THE DRAWINGS:

Please replace the drawing sheet currently on file in this application with the enclosed Replacement Sheet for Figures 1, 2 and 3.

REMARKS

In the office Action, the Examiner rejected Claims 1-21, which were all of the then pending claims, under 35 U.S.C. 103 as being unpatentable over a Research Report titled "A Software Falsifier" (Brand) in view of U.S. Patent 6,901,581 (Schneider). The Examiner also objected to Figures 1-3 because they are not designated as "Prior Art."

Applicants are herein amending independent Claims 1, 9 and 16 to better define the subject matters of these claims. New Claims 22 and 23, which are dependent from Claim 1, are being added to describe preferred or optional features of the present invention, and Claims 2, 4, 10 and 17 are being cancelled to reduce the number of issues in this application.

Also, Applicants are submitting herewith-amended Figures 1-3, in which these Figures are labeled "Prior Art." In view of these amended Figures, the Examiner is asked to reconsider and to withdraw the objection to the drawings.

In addition, for the reasons explained below, Claims 1, 3, 5-9, 11-16 and 18-23 patentably distinguish over the prior art and are allowable. The Examiner is thus asked to reconsider and to withdraw the rejection of Claims 1, 3, 5-9, 11-16 and 18-21, and to allow these claims and new Claims 22 and 23.

The present invention, generally, relates to identifying errors in computer programs, and more particularly to determining whether suspected errors are true errors. As discussed in the present application, there are several known methods for identifying errors in software programs, including testing, program verifiers, and static analysis tools. A program verifier, for example, attempts to prove that a formal assertion of correct behavior is true for all possible inputs and for all possible execution paths. Weak static analysis tools use syntactic and data-flow analysis techniques found in computers to identify possible program errors.

Strong static analysis tools use additional theorem proving and symbolic execution techniques to reduce the number of false error reports and to specify the program conditions that lead to an erroneous program execution.

While there is a wide range of existing tools to identify program error, each approach has its limitations. Weak static analysis tools typically produce a large number of false error reports. The strong static analysis tools reduce the number of false error reports; however, false error reports are still a possibility.

The present invention addresses the issue of false error reports. Generally, this is done by analyzing a computer program to generate an initial error report and a list of conditions for suspected errors. A set of assertions are generated and inserted into the computer program. The computer program is then re-executed, with the assertions therein, while monitoring for the conditions for the suspected errors. If the conditions for one of the suspected errors are satisfied, during the re-execution of the computer program, a second error report is generated indicating that said one of the suspected errors is a true error.

The prior art does not disclose or suggest confirming whether a suspected error is a true error, in the manner described above.

For instance, Brand, the primary reference relied on by the Examiner, describes a strong static analysis tool. This reference was cited by the Applicant and discussed in the present application. This tool is very useful. However, as explained in the instant application, these tools tend to take a conservative approach to reduce the number of false error reports but at the expense of ignoring some potential errors. When the tool disclosed in Brand cannot determine if a particular error will occur in actual program execution, the tool chooses not to report the potential error.

The present invention takes a different approach. The instant invention identifies these potential errors, and then re-executes the program with the assertions inserted, to determine if the conditions occur that would cause the suspected error to occur.

Schneider discloses a procedure for debugging a computer program and in which a simulated execution is used to progressively isolate the bug. In this procedure, program information is recorded in a trace buffer. This program information generally includes write accesses from the computer program and branches in an execution path of the program. Also, a memory image, referred to as a snapshot," of a portion of memory is captured.

Execution of the computer program is then simulated by changing the memory image. This simulation involves execution of the information recorded in the trace buffer and the memory image. Importantly, as Schneider acknowledges in column 5, lines 56-58, during this simulation, the "original code is not actually being re-executed." In addition, it is significant to note that, as Schneider also acknowledges, in column 4, lines 24 and 25, the procedure disclosed therein "involves using an enormous trace buffer."

There are important differences between the present invention and the procedure disclosed in Schneider. One important difference is that Schneider is directed to isolating errors by making those errors repeatable, while the present invention is directed to confirming whether or not suspected errors are true errors. Another important difference between the present invention and Schneider is that, with the present invention, the original computer program is re-executed, while in the Schneider procedure, the original program is not re-executed.

Independent Claims 1, 9 and 16 are being amended to better describe the above-discussed differences. More specifically, as presented herewith, Claims 1 and 16 describe the steps of re-executing the computer program, with the assertions therein, while monitoring for the conditions for the suspected errors; and if, during said re-executing step, the conditions for one of the suspected errors are satisfied then generating a second error report indicating that said one of the suspected errors is a true error.

Claim 9 is directed to a system for analyzing a computer program and this claim describes analogous system features. In particular, Claim 9 describes a program executor for re-executing the computer program, with the assertions therein, while monitoring for the conditions for the suspected errors; and if, during said re-executing step, the conditions for one of the suspected errors are satisfied, for then generating a second error report indicating that said one of the suspected errors is a true error.

The systems disclosed in Brand and Schneider do not operate in this way. In particular, Schneider detects error. In comparison, the present invention, during the re-execution of the computer program, does not monitor for the errors, but instead monitors for the error conditions.

For instance, the example code shown in Figures 6 and 7 of the present application has a step L1 at which A is divided by X. An error occurs at this step if X is equal to 0. The Schneider procedure might identify this error at step L1. This is not what the present invention does during the re-execution of the computer program. Instead, during that re-execution, the present invention monitors for the condition X=0, which would occur earlier in the program and which would verify that the suspected error (possible division by zero) is in fact a true error. When this condition (X=0) occurs, the present invention generates a report

indicating that the suspected error is a true error.

The present invention is of utility for a number of reasons. As discussed in the present application, the invention helps to find real errors and reduces or eliminates false error reports. Moreover, this is done without needing the "enormous" buffer required by Schneider.

The other references of record have been reviewed, and these other references, whether considered individually or in combination, also do not disclose or suggest the above-discussed features of the present invention.

Because of the above-discussed differences between Claims 1, 9 and 16 and the prior art, and because of the advantages associated with these differences, Claims 1, 9 and 16 patentably distinguish over the prior art and are allowable. Claims 3, 5-8, 22 and 23 are dependent from Claim 1 and are allowable therewith. Similarly, Claims 11-15 are dependent from Claim 9 and are allowable therewith; and Claims 18-21 are dependent from, and are allowable with, Claim 16. The Examiner is, accordingly, respectfully requested to reconsider and to withdraw the rejection of Claims 1, 3, 5-9, 11-16 and 18-21, and to allow these claims and new Claims 22 and 23.

In light of the above-discussion, the Examiner is asked to reconsider and to withdraw the objection to the Drawings, to reconsider and to withdraw the rejection of Claims 1, 3-9, 11-16 and 18-21 under 35 U.S.C. 103, and to allow these claims and Claims 22 and 23.

If the Examiner believes that a telephone conference with Applicants' Attorneys would be advantageous to the disposition of this case, the Examiner is requested to telephone the undersigned.

Respectfully submitted,

John S. Sensny

John S. Sensny
Registration No.: 28,757
Attorney for Applicants

Scully, Scott, Murphy & Presser, P.C.
400 Garden City Plaza – Suite 300
Garden City, New York 11530
(516) 742-4343

JSS:jy
Enclosures: One Replacement Sheet and One Annotated Sheet for Figures 1, 2 and 3

ANNOTATED SHEET

FIG. 1

PRIOR ART

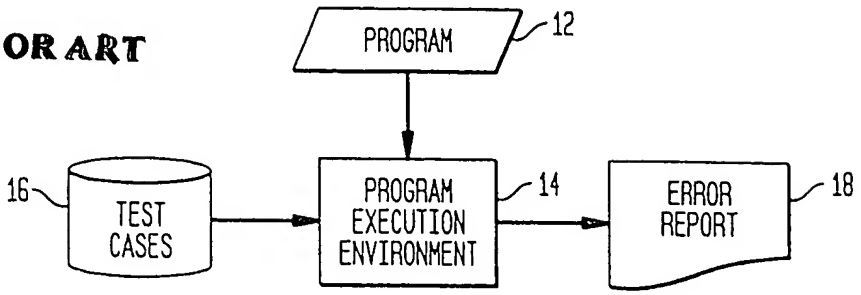

FIG. 2

PRIOR ART

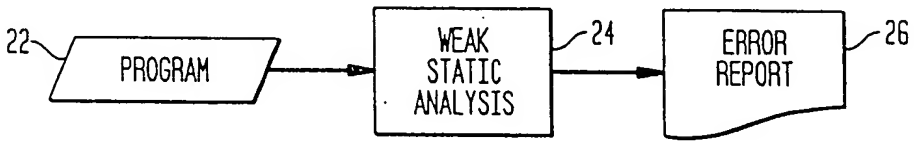

FIG. 3

PRIOR ART